Designing Optimal Curves in 2D

Stefan Gumhold
LSI/ETSII
University of Granada
e-mail: gumhold@gris.uni-tuebingen.de

Abstract

The use of non-linear optimal curves for an intuitive design with interpolating curves is proposed. The curve design system is based on an optimization algorithm that can minimize a variety of optimality functionals, which are based on the integration of the curve length, curvature and curvature derivatives. Besides the to be interpolated points further constraints on the curve normals can be incorporated easily into the optimization approach. It is furthermore shown how to design interpolating curves with continuity higher than C^1.

Keywords: Interpolation, Optimal Curves, Curve Design

1 Introduction

Linear and rational splines are widely used in design and drawing tools. Their invention dates back to an early paper of Schoenberg [19] and a comprehensive treatment can be found first in [1] and later in [3]. For a curve design with interpolating curves [4], it is however very hard to avoid unnecessary loops and wiggles. In this paper we propose to solve the interpolation problem with optimal curves that minimize an user defined functional. The functional is defined from the curve length, curvature and curvature derivatives and includes parameters that can be adjusted by the designer to fine-tune the shape of the curve.

The interpolation with optimal curves has been studied in the literature since the sixties. The research has been motivated by the ship-building, aircraft and car industry, where the technique of lofting was employed to design shapes. For this thin wooden planks were passed through points laid out on the floor of a large design loft. The physical model for the thin wooden plank is the elastic line, which tries to minimize its internal energy. The internal energy is by the Euler-Bernoulli law [21] proportional to the integral over the squared curvature of the elastic line (see also [13]). If the elastic line is represented as a curve $\mathbf{c}(s) : \Re \rightarrow \Re^2$,

the functional of the integrated curvature reads

$$K(\mathbf{c}) = \int_{s=0}^{L(\mathbf{c})} \kappa^2(s) ds. \tag{1}$$

The curve is assumed to be parameterized by its arc length s and $L(\mathbf{c})$ is the total length of the curve. In 2d the curvature $\kappa(s)$ is simply $det(\mathbf{c}', \mathbf{c}'')$ with the first $\mathbf{c}'$ and second $\mathbf{c}''$ derivatives for the arc length s, where $\|\mathbf{c}'(s)\| \equiv 1$.

If the elastic line is constraint to pass through a given list $\mathbf{p}_i$ of interpolation points, it minimizes the internal energy or equivalently 1 under the given interpolation constraints, where it is typically assumed that the length of the elastic line is not constrained. In order to compute the shape of the elastic line, one has to solve for each segment S_i between $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ the minimization problem

$$\mathbf{c}_i = \underset{\mathbf{c}:\mathbf{c}(0)=\mathbf{p}_i \wedge \mathbf{c}(L(\mathbf{c}))=\mathbf{p}_{i+1}}{\operatorname{minarg}} K(\mathbf{c}).$$

The solution curves $\mathbf{c}_i$ are the so-called non-linear splines and can be computed by plugging $f(s) = \kappa^2(s)$ into the Euler-Lagrange equations. This yields [13] the following second order differential equation in the curvature

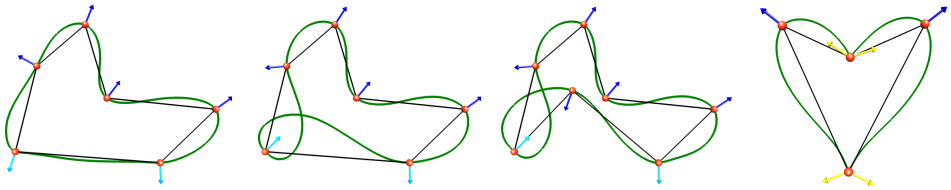
$$\kappa'' + \frac{1}{2}\kappa^3 = 0, \tag{2}$$

where the derivatives are with respect to the arc length parameter s. This equation can be solved in a cylindrical coordinate system, where it transforms into a simple wave equation in the square of the curvature.

If the variational function $f(s)$ depends on the $n-$th derivative of the curve $\mathbf{c}(s)$ and on no higher derivatives, boundary conditions for $\mathbf{c}(0), \mathbf{c}'(0), \ldots, \mathbf{c}^{(n-1)}(0)$ and for $\mathbf{c}(L(\mathbf{c}))$, $\mathbf{c}'(L(\mathbf{c}))$, ..., $\mathbf{c}^{(n-1)}(L(\mathbf{c}))$ are necessary to specify the optimal curve uniquely. In the case of the non-linear spline $f(s)$ depends on $\kappa(s)$ and therefore on the second derivative of $\mathbf{c}(s)$, such that besides the interpolation constraints also the first derivative of the end points of each segment have to be constrained to uniquely define the non-linear spline. At interior points $\mathbf{p}_i$ the incident spline segments are typically constrained to fulfill the C^1-continuity condition $\mathbf{c}'_{i-1}(\mathbf{p}_i) = \mathbf{c}'_i(\mathbf{p}_i)$, whereas at free end points the natural boundary condition is chosen, that enforces zero curvature.

The non-linear spline interpolation problem has been solved in different ways [11, 14, 6, 13, 17, 9, 10, 7]. One can distinguish two general types of approaches. In the direct approach one tries to optimize the curve directly, whereas in the indirect approach one first seeks for a solution to the differential equation 2 and then determines a curve with the given curvature values. In the indirect approach one typically has to switch several times between solving the curvature DGL and embedding the curve (see for example [18]).

Besides the physically founded curvature based functional 1 a variety of other to be minimized functionals have been investigated for the design of curves. Mächler [12] minimizes



a) b) c) d)

Figure 1: Sample interaction with the system. The red balls are the interpolation constraints. Dark blue arrows show the curve normals resulting from the optimization. The cyan arrows are normal constraints specified by the user and the yellow arrows are one sided normal constraints. a) → b) change of normal constraint; b) → c) insertion of point. d) a heart designed with two double normal constraints, that introduce two sharp corners.

the square of the relative curvature change κ'/κ to approximate data points (x_i, y_i) with a smooth function. The appearance of κ in the denominator penalizes reflection points (points, where the curvature becomes zero) with an infinite penalty, such that the number of reflection points has to be specified by the user. Moreton and Sequin [16] use only the squared derivative κ' of the curvature to define optimal interpolating curves. Schneider and Kobbelt [18] use the simplified version $\kappa'' = 0$ of the differential equation 2 to define a fair interpolating curve, as this DGL can be generalized to surfaces. Alon and Bergmann [2] analytically solved the variational minimization problem for arbitrary curvature exponents α, i.e.

$$\mathbf{c}_i = \underset{\mathbf{c}:\mathbf{c}(0)=\mathbf{p}_i \wedge \mathbf{c}(L(\mathbf{c}))=\mathbf{p}_{i+1}}{\operatorname{minarg}} \int_{s=0}^{L(\mathbf{c})} |\kappa(s)|^{\alpha} ds.$$

The resulting analytic expressions of the optimal curves can be expressed in terms of transcendental functions and make it quite complicated to combine several spline segments continuously.

In this work a curve design system is developed, which is based on a general optimization algorithm that can handle a wide variety of functionals based on the length of the curve, its curvature and the curvature derivatives. The user specifies a polygon $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n$ that represents the ordered interpolation constraints. He can insert, remove and move the points with simple mouse interaction. Furthermore he can scale and rotate the design plane. At each point $\mathbf{p}_i$ additional constraints on the curve normal can be specified. Figure 1 shows an example interaction and a heart with double normal constraints set, i.e. different normals for the incoming and outgoing curve.

For curve design it turned out that the combination of curve length and curvature results in an adjustable functional, which is very intuitive and allows to trade-off between smooth and short curves as detailed in section 3. This is very similar to exponential splines derived from a linear differential ansatz [20]. But before that, the curve optimization algorithm is

explained in section 2. The main contributions of the paper are

- a curve design system based on optimal curves
- an adaptive curve optimization algorithm supporting a very general optimization functional
- a new discrete curvature measure that significantly improves convergence during optimization
- an intuitive curve design paradigm based on length and curvature
- a proposal for the generation of interpolating C^∞ curves

2 Curve Optimization

This section describes the proposed definition of optimal curves and how these are computed. The functional itself is presented in 2.1. It is very general and contains most known functionals as a special case. One important design criterion hereby is that the minimizing curves have to be not only rotational and translational invariant, but also independent of scale.

As the Euler Lagrange equations of the proposed functional become too complicated to be solved analytically, an optimization algorithm is used to minimize the functional directly on a sequence of polygons that approximate the optimal curve with increasing resolution. The curve discretization is detailed in section 2.2. The optimization algorithm as described in 2.3 is applied hierarchically and adaptively.

2.1 Optimality Functional

It is well known that the curvature is invariant under rotations and translations. The same holds for the length of the curve and the infinitesimal arc length element ds. Neither of the two functionals are invariant under scaling. If a curve $\mathbf{c}$ is scaled by a factor of q to $\tilde{\mathbf{c}} = q\mathbf{c}$ the length of the curve is also scaled by q, but the curvature by $\frac{1}{q}$. This is because the curvature is proportional to the inverse of the bending radius, which is scaled by q. In general do the curvature and its derivatives scale as

$$\mathbf{c} \longrightarrow q\mathbf{c} \qquad \kappa \longrightarrow \frac{1}{q}\kappa \qquad \kappa^{(k)} \longrightarrow \frac{1}{q^{k+1}}\kappa^{(k)}. \tag{3}$$

The different behavior under scaling is not a problem for the definition of optimal curves based only on curvature or only on length. The minimal value of the functional does change under scaling, but the minimizing curve stays the same as the scaling factor q can be taken out of the integral:

$$\int \left(\frac{\kappa}{q}\right)^2 q\,ds = \frac{1}{q}\int \kappa^2 ds. \tag{4}$$

When combining the curve length, curvature and curvature derivatives into a joined functional, care has to be taken that the minimizing curve does not dependent on the scale.

In the proposed approach the curvature and its derivatives are multiplied with a property that has the reciprocal dependence on the scale. The simplest property, which is proportional to the scale, is the length L_0 of the polygon formed by the interpolation constraints $\mathbf{p}_i$. Therefore, the scale independent optimization functional is defined as

$$F(\mathbf{c}, \beta, \beta_0, \ldots, \beta_d, \alpha_0, \ldots, \alpha_d) \quad = \quad \int_{s=0}^{L(\mathbf{c})} \beta + \beta_0 \left|\kappa L_0\right|^{\alpha_0} \quad + \quad \sum_{k=1}^{d} \beta_k \left|\kappa^{(k)} L_0^{k+1}\right|^{\alpha_k} ds, \tag{5}$$

with the specializations $L(\mathbf{c}) = F(\mathbf{c}, 1, 0, \ldots)$ and $K(\mathbf{c}) = F(\mathbf{c}, 0, 1, 0, \ldots, 0, 2/L_0^2, 0, \ldots)$. It is easy to check that $F(q\mathbf{c}) = qF(\mathbf{c})$, what implies the invariance of the minimizing curve under scaling. All the αs and βs can be adjusted in the proposed design system.

2.2 Discretization

2.2.1 Polygonal Approximation

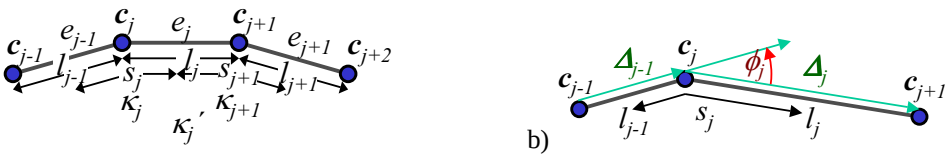


Figure 2: Notation for the discretization of the optimal curve into polygons.

The curve is approximated by a discrete polygon $\mathbf{c}_0, \ldots, \mathbf{c}_j, \ldots$ on each resolution level during hierarchical curve optimization. Figure 2 a) illustrates the used notation. The subscript j is used to distinguish the entities of the approximating polygon from the entities of the polygon $\mathbf{p}_i$ that describes the interpolation constraints. For each edge e_j the length is denoted by l_j. At each point $\mathbf{c}_j$ the lengths of the incident edges are averaged to the incremental arc length s_j.

2.2.2 Curvature Discretization

To be able to evaluate the functional 5 also the discrete curvatures κ_j have to be approximated, what is done at each point $\mathbf{c}_j$. For this the delta vectors $\mathbf{\Delta}_j = \mathbf{c}_{j+1} - \mathbf{c}$ are defined as illustrated in Figure 2 b). A commonly used curvature discretization results from fitting a circle through $\mathbf{c}_{j-1}, \mathbf{c}_j$ and $\mathbf{c}_{j+1}$ and from using the inverse radius as a curvature approximation κ°. One can also use the discrete change ϕ_j (see Figure 2 b) of the tangent together with

the curvature definition $\kappa(s) = \partial\phi/\partial s$ in order to come up with a second discretization κ^ϕ. Given the normalized delta vectors $\mathbf{c}'_j = \boldsymbol{\Delta}_j / l_j$ the two curvature discretizations are

$$\kappa_j^\circ = \frac{\det(\mathbf{c}'_{j-1}, \mathbf{c}'_j)}{||\mathbf{c}_{j+1} - \mathbf{c}_{j-1}||} = \frac{\sin\phi_j}{||\mathbf{c}_{j+1} - \mathbf{c}_{j-1}||} \quad \text{and} \quad \kappa_j^\phi = \frac{\phi_j}{2s_j}. \tag{6}$$

Both curvature approximations are useful for small angles ϕ_j. As the input to the curve

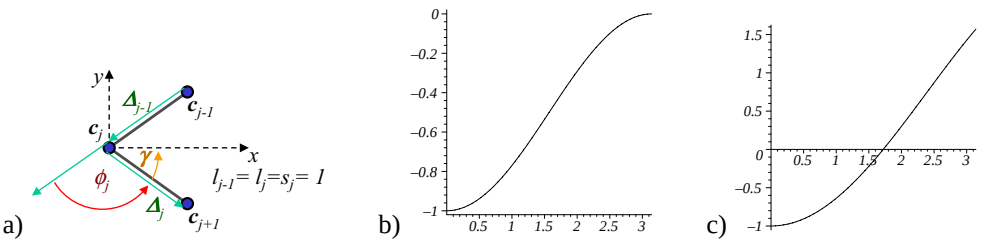


Figure 3: Examination of curvature gradient in dependence of tangential change ϕ_j: a) examined scenario: partial derivative of κ_j with respect to the x-component of $\mathbf{c}_j$; b) plot of $\frac{\partial}{\partial x}\kappa_j^\circ(\phi_j)$; c) plot of $\frac{\partial}{\partial x}\kappa_j^\phi(\phi_j)$.

design system is an arbitrary user defined polygon $\mathbf{p}_i$, all angles in the range of $\phi_j \in [-\pi, \pi]$ can arise. In Figure 3 the problem with using the standard curvature measures is illustrated in the simplified situation shown in a). For the optimization algorithm the crucial ingredient is the gradient of the curvatures κ_j for the $\mathbf{c}_j$. By symmetry is the major curvature change in Figure 3 a) along the x-direction. The simple geometry $l_{j-1} = l_j = s_j = 1$ allows us to compute the partial derivatives of κ_j° and κ_j^ϕ with respect to the x-component of the point $\mathbf{c}_j$. The resulting plots are shown in Figure 3 b) and c). The gradient of κ_j° is negative, which implies that the curvature increases with a displacement of $\mathbf{c}_j$ in negative x-direction. This makes sense as the tangential angle change ϕ_j increases. But with increasing ϕ_j the gradient of κ_j° decreases to zero. This is not what one would expect and also yields to a convergence problem for the optimization procedure that has to get rid of all sharp angles eventually to generate a smooth curve. For the discretization κ_j^ϕ the gradient even becomes positive for angles $\phi_j > \pi/2$.

As both curvature discretizations are a function of ϕ_j over a length, we start with the general ansatz

$$\kappa_j^f(x) = \frac{f(\phi_j)}{2s_j} = \frac{f\left(\pi - 2\mathrm{atan}\left(\frac{\sin\gamma}{\cos\gamma - x}\right)\right)}{2\sqrt{\sin^2\gamma + (\cos\gamma - x)^2}}, \tag{7}$$

where x denotes the displacement of the x-coordinate of $\mathbf{c}_j$ and $\gamma = (\pi - \phi_j)/2$ the inner half-angle as labeled in Figure 3 a). A displacement of $\mathbf{c}_j$ by x along the x-axis reduces $\cos\gamma$ by x, which is accounted for in the above formula on the right with the help of the arc tangent function.

Some algebra yields the derivative of 7 with respect to x at $x = 0$, which corresponds to the gradient:

$$\frac{\partial}{\partial x}\kappa_j^f(x=0) = \frac{1}{2}f(\phi_j)\cos\gamma - \frac{\partial f}{\partial \phi_j}(\phi_j)\sin\gamma = \frac{1}{2}f(\phi_j)\sin\frac{\phi_j}{2} - \frac{\partial f}{\partial \phi_j}(\phi_j)\cos\frac{\phi_j}{2}. \quad (8)$$

Any desired gradient can now be designed with an appropriate choice of f. Here the gradient was set to a constant $-g$, but any other choice would have been possible. The result is a differential equation for $f(\phi_j)$:

$$0 = \frac{1}{2}f(\phi_j)\sin\frac{\phi_j}{2} - \frac{\partial f}{\partial \phi_j}(\phi_j)\cos\frac{\phi_j}{2} + g,$$

with the solution $f(\phi_j) = (g\phi_j + \Phi)/\cos\frac{\phi_j}{2}$, where Φ is an integration constant. Φ is fixed to be zero and g to be one by the additional constraint that κ has to behave around $\phi_j = 0$ like $\phi_j/2s_j$. The solution $f(\phi_j) = \phi_j/\frac{\cos\phi_j}{2}$ goes to infinity when the inner angle $2\gamma \to 0$, i.e. $\phi_j \to \pm\pi$. This makes intuitively a lot of sense as an arbitrarily sharp corner implies an infinite curvature, but in praxis the infinite value is hard to handle.

By inspection it can be shown furthermore that $f(\phi_j) = \phi_j/\cos\frac{\phi_j}{2+\delta}$ results for all $0 < \delta < 1$ in a constant gradient between minus one and zero. As $|\phi_j|$ is less than π, the modified function maps all $\phi_j \in [-\pi, \pi]$ to finite values. If for example $\delta = 0.1$ is chosen, no values larger than 21.1 will arise. Putting it all together, the following discretization of the curvature is proposed

$$\kappa_j(s_j, \phi_j) = \frac{\phi_j}{2s_j\cos\frac{\phi_j}{2.1}}. \quad (9)$$

2.2.3 Derivatives and Integration

Finally, the curvature derivatives and the integral in the definition of functional 5 have to be discretized. The derivatives were simply computed with one-sided finite differences. Every odd order derivative can be thought of being computed at the edge centers, whereas the even order derivatives are computed at the points $\mathbf{c}_j$ like the curvature itself. The discrete curvature derivatives are defined recursively from the curvatures κ_j, which are identified with $\kappa_j^{(0)}$

$$\begin{aligned} \forall \text{ odd } k \geq 1: \quad \kappa_j^{(k)} &= \left(\kappa_{j+1}^{(k-1)} - \kappa_j^{(k-1)}\right)/l_j \\ \forall \text{ even } k \geq 2: \quad \kappa_j^{(k)} &= \left(\kappa_j^{(k-1)} - \kappa_{j-1}^{(k-1)}\right)/s_j \end{aligned}$$

Notice that the arc length element s_j is used for derivatives defined at points and the edge lengths l_j for derivatives on edge centers. Similarly, s_j and l_j respectively have to be multiplied for integration

$$\begin{aligned} &F_{\text{discrete}}(\mathbf{c}, \beta, \beta_0, \ldots, \beta_d, \alpha_0, \ldots, \alpha_d) \quad = \\ &\sum_j \left[\left(\beta + \sum_{k=1,3,\ldots} \beta_k \left| \kappa^{(k)} L_0^{k+1} \right|^{\alpha_i} \right) l_j \quad + \quad \left(\beta_0 \left| \kappa_j L_0 \right|^{\alpha_0} + \sum_{k=2,4,\ldots} \beta_k \left| \kappa^{(k)} L_0^{k+1} \right|^{\alpha_i} \right) s_j \right]. \end{aligned}$$

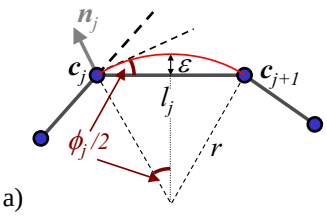
2.3 Hierarchical and Adaptive Optimizer

The optimal curve is approximated by a sequence $\mathbf{c}_j^0, \mathbf{c}_j^1, \ldots$ of polygons with increasing resolution and decreasing edge length, that converges to the minimizing curve $\mathbf{c} = \lim_{a \to \infty} \mathbf{c}_j^a$. The polygon $\mathbf{p}_i$ that connects the interpolation constraints is used as starting point $\mathbf{c}_j^0$, from which also the length L_0 is computed. This is a good starting point as $\mathbf{c}_j^0$ minimizes the total length $L(\mathbf{c})$, i.e. $F(\mathbf{c}, 1, 0, \ldots)$.

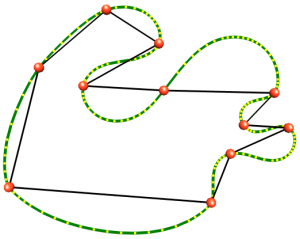
The next finer approximation is derived from $\mathbf{c}_j^0$ by adding a vertex in the middle of each edge. Other center placement strategies were tried but proved to be less stable.

2.3.1 Adaptivity

The user can specify a target resolution ϱ as termination criterion for the edge subdivision process. The resolution is measured in pixels per bounding box extent of the curve. This allows to compute a maximum allowed error $\epsilon_{\max}$. Instead of subdividing all edges, only those that cause an error larger than $\epsilon_{\max}$ are subdivided.



a)



b)

Figure 4: a) maximal edge error is in the middle of the edge and computed from $r \cos \phi_j/2 = l_j/2$ and $r \sin \phi_j/2 = 1 - \epsilon$; b) example of an adaptively subdivided curve with a target resolution of $\varrho = 1000$.

The approximation error at an edge of the polygon depends on the length l_j of the edge and the incident angles ϕ_j and ϕ_{j+1}. Each of the angles ϕ_j defines together with the length l_j a circle that is tangential to the curve at $\mathbf{c}_j$ with relation to the approximated normal $\mathbf{n}_j$. Figure 4 a) shows that the maximum error at the edge center computes to

$$\epsilon(l_j, \phi_j) = l_j \left(1 - \cos \frac{\phi_j}{2}\right) / 2 \sin \frac{\phi_j}{2}.$$

In each subdivision step all edges with $\max\{\epsilon(l_j, \phi_j), \epsilon(l_j, \phi_{j+1})\} > \epsilon_{\max}$ are subdivided. Figure 4 b) shows an example of an adaptively subdivided curve.

2.3.2 Conjugate Gradient Minimization

After each subdivision a conjugate gradient minimization algorithm is used to place the unconstraint points in order to minimize the functional F. The movement of each point was restricted to the direction normal to the curve by reducing the $2n$ variables $\mathbf{c}_j$ to the n variables δ_j, with $F(\delta_j) = F(\mathbf{c}_j + \delta_j \mathbf{n}_j)$ and the normal direction $\mathbf{n}_j$. To define the normal direction, the operator $*$ is introduced to define the rotation of a vector by $\pi/2$

$$\begin{pmatrix} x \\ y \end{pmatrix}^* := \begin{pmatrix} y \\ -x \end{pmatrix}.$$

The normal directions can then be defined from the normalized tangent vectors as

$$\mathbf{n}_j = \frac{(\mathbf{c}'_{j-1} + \mathbf{c}'_j)^*}{||\mathbf{c}'_{j-1} + \mathbf{c}'_j||}.$$

The main ingredient to the conjugate gradient minimizer is a method that computes the gradient

$$\nabla_{\delta_j} F(\mathbf{c}_j + \delta_j \mathbf{n}_j) = (F_{\delta_1}, \ldots, F_{\delta_n})\,(\mathbf{c}_j + \delta_j \mathbf{n}_j),$$

with F_{δ_j} being the partial derivative of F for δ_j.

When the normal direction is kept fix, this gradient can be computed from the gradient matrix for the $\mathbf{c}_j = (x_j, y_j)$ consisting of all the 2d partial derivatives for $\mathbf{c}_j$

$$\nabla_{\mathbf{c}_j} F(\mathbf{c}_j) = (F_{\mathbf{c}_1}, \ldots, F_{\mathbf{c}_n})\,(\mathbf{c}_j) = \left(\begin{bmatrix} F_{x_1} \\ F_{y_1} \end{bmatrix}, \ldots, \begin{bmatrix} F_{x_n} \\ F_{y_n} \end{bmatrix} \right)(\mathbf{c}_j)$$

with the scalar products $F_{\delta_j} = (F_{x_j} \;\; F_{y_j})\mathbf{n}_j$.

The gradient matrix $\nabla_{\mathbf{c}_j} F$ was computed analytically in order to avoid numerical problems. The main constitutes are the 2d partial derivatives of the length element s_j and the curvature κ_j for the $\mathbf{c}_k$ with $k = j-1, j, j+1$. The angle ϕ_j in the curvature definition can be written in terms of the delta vectors as

$$\phi_j = \operatorname{atan} \frac{Y_j}{X_j}, \quad X_j := \Delta_{j-1}^T \Delta_j, \quad Y_j := \Delta_{j-1}^T \Delta_j^*.$$

The curvature derivatives can be expressed with $\rho_j := \phi_j / 2.1$ and $f(\phi_j) := \phi_j / \cos\rho_j$ as

$$\frac{\partial \kappa_j}{\partial \mathbf{c}_k} = \left[\frac{1}{cos\rho_j} + \frac{f(\phi_j)\tan\rho}{2.1} \right] \frac{X_j \frac{\partial Y_j}{\partial \mathbf{c}_k} - Y_j \frac{\partial X_j}{\partial \mathbf{c}_k}}{2 s_j (X_j^2 + Y_j^2)} - \frac{\kappa_j}{s_j} \frac{\partial s_j}{\partial \mathbf{c}_k}.$$

Finally, the partial derivatives of X_j, Y_j and s_j for $k = j-1, j, j+1$ can be written in vector notation with the upper component corresponding to $k = j-1$ and the lower to $k = j+1$

$$\frac{\partial X_j}{\partial \mathbf{c}_k} = \begin{Bmatrix} -\Delta_j \\ \Delta_j - \Delta_{j-1} \\ \Delta_{j-1} \end{Bmatrix}, \quad \frac{\partial Y_j}{\partial \mathbf{c}_k} = \begin{Bmatrix} -\Delta_j^* \\ \Delta_{j-1}^* + \Delta_j^* \\ -\Delta_{j-1}^* \end{Bmatrix}, \quad 2\frac{\partial s_j}{\partial \mathbf{c}_k} = \begin{Bmatrix} -\mathbf{c}'_{j-1} \\ \mathbf{c}'_{j-1} - \mathbf{c}'_j \\ \mathbf{c}'_j \end{Bmatrix}.$$

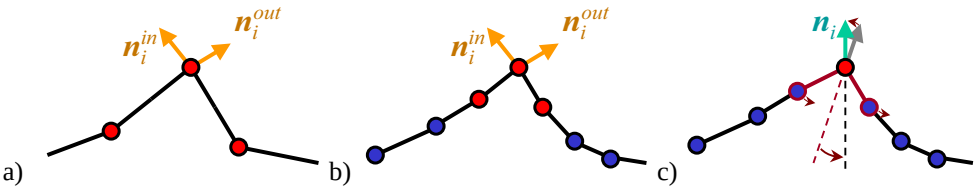


Figure 5: Incorporation of normal constraints into hierarchical optimization: a) for constraint with in- and outgoing normals, adjacent points are also fixed; b) after subdivision again the adjacent points are fixed by normal constraint; c) single normal constraints are kept by rotating back the two adjacent points to align computed normal with constraint normal.

All derivatives of the functions in the functional that depend on κ_j and s_j can be derived from the given formulas with the basic derivative rules.

To incorporate the interpolation constraints, the δ_j of the points $\mathbf{c}_j$, that coincided with an interpolation constraint $\mathbf{p}_i$ were set to zero.

With the known gradient the step width σ necessary to the conjugate gradient algorithm was determined by minimizing the one parameter function $F(\sigma) = (\mathbf{c}_j + \sigma\delta_j\mathbf{n}_j)$ with a maximum of ten Newton-Raphson iterations. After performing each step $\mathbf{c}_j \leftarrow \mathbf{c}_j + \sigma\delta_j\mathbf{n}_j$ the normals and the δ_j-gradients were recomputed. The conjugate gradient optimization typically converged for each resolution $\mathbf{c}_j^a$ after between ten and twenty steps allowing for an interactive curve design with about two curve optimizations per second on a P4 with 2GHz for a target resolution of 500 pixels.

3 Curve Design

3.1 Adding Constraints

In a lot of situations the user wants to explicitly specify the curve normals and introduce sharp corners. These additional constraints can be easily incorporated into the design system. At the moment two further constraints on the normals are supported. The user can either specify the normal direction $\mathbf{n}_i$ for a given interpolation constraint $\mathbf{p}_i$, or he can specify the incoming normal $\mathbf{n}_i^{\text{in}}$ and the outgoing normal $\mathbf{n}_i^{\text{out}}$. In Figure 1 the normal constraints are visualized by cyan arrows and the double normal constraints with two yellow arrows.

As non of the normal constraints has to be fulfilled in the initial polygon $\mathbf{c}_j^0$, an new initial polygon $\tilde{\mathbf{c}}_j^0$ was computed, where all normal constraints are satisfied. For this all edges incident to a single or double normal constraint were split and the newly inserted point were positioned in a way to fulfill the corresponding normal constraint.

In order to keep all normal constraints satisfied during optimization, all points incident to a double normal constraint were fixed as illustrated in Figure 5 a). After each subdivision, the previously fixed neighbors become free and the new direct neighbors are fixed (see Fig-

ure 5 b). The curvatures at the point with the double normal constraints was set to zero to implement natural boundary conditions.

The single normal constraint is more difficult to fulfill, at least if the curvature is not fixed. The single normal constraint is accomplished by updating the neighboring points after each addition of the gradient. For this the actual normal was computed and the two direct neighbors were rotated around the constrained point such that the current normal matches the constraint normal (see Figure 5 c). In this way the curvature can change freely. It would also be easy to add constraints on the curvatures themselves. But this is not yet implement.

3.2 Trading Off Between Length and Curvature

So far it was only discussed how the user can specify point and normal constraints, but the free parameters of the functional 5 have not been exploited yet. The most intuitive contributions to the functional are the total length of the curve, which is weighted by β in F, and the total curvature, which is taken to the power of α_0 and weighted by β_0. Let us therefore first examine the restriction of the functional to $F(\mathbf{c}, \beta, \beta_0, 0, \ldots, \alpha_0, 0, \ldots)$. Some experimentation showed that the influence of the curvature exponent α_0 is not very intuitive to design a curve and it was simply fixed to the physically motivated value of $\alpha_0 = 2$.

The two parameters β and β_0 remain to be adjusted by the user. This is one parameter too much as the curve that minimizes the functional does not change if the functional is multiplied by a constant. The simplest approach would be to set one parameter to one, for example $\beta = 1$. The other parameter β_0 would have to be varied between zero and infinity to generate all possible minimizing curves.

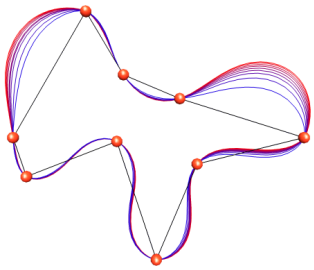
For the user it is easier to trade off between a short and a low-curvature curve with a parameter $\lambda \in [0, 1]$, where $\lambda = 0$ corresponding to the shortest possible curve and $\lambda = 1$ to the curve with the minimal curvature. Again the simplest approach of setting $\beta = 1 - \lambda$ and $\beta_0 = \lambda$ does not provide an intuitive control for the user. This will be explained right after introducing the solution to the problem that uses λ and a to be determined scale factor q

$$F_{simp}(\mathbf{c}, \lambda) = \int_{s=0}^{L(\mathbf{c})} (1 - \lambda)q + \lambda \left|\kappa L_0\right|^2 ds, \tag{10}$$

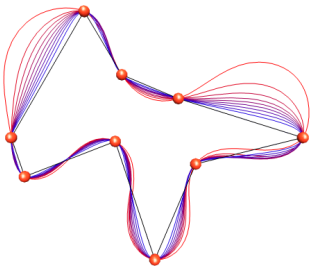
with $\beta = q(1 - \lambda)$ and $\beta_0 = \lambda$. In order to understand why the scale factor q is necessary, let us define $\mathbf{c}_{simp}(\lambda)$

$$\mathbf{c}_{simp}(\lambda) = \underset{\mathbf{c}:\mathbf{c}(x_i)=\mathbf{p}_i}{\operatorname{minarg}} F_{simp}(\mathbf{c}, \lambda)$$

as the curve that minimizes the simplified functional. The curve $\mathbf{c}_{simp}(1)$ is clearly independent of q as the corresponding functional does not contain the length term. But also $\mathbf{c}_{simp}(0)$ is independent of q, even if q varies along the curve. It will always be the polygon connecting the interpolation constraints $\mathbf{p}_i$. This is also the reason, why q was incorporated into β and not into β_0, what would have changed the curve $\mathbf{c}_{simp}(1)$ if q varies over the curve. The curves $\mathbf{c}_{simp}(0)$ and $\mathbf{c}_{simp}(1)$ are used as reference curves to determine q.



a)



b)

Figure 6: Influence of q on the uniformity of the λ control to trade off between short and low-curvature curves: a) and b) show the minimizing curves for $\lambda = 0, 0.1, \ldots, 1$; a) $q = 100$: most λ-curves are nearly identical; b) $q = 5000$: much more uniform distribution.

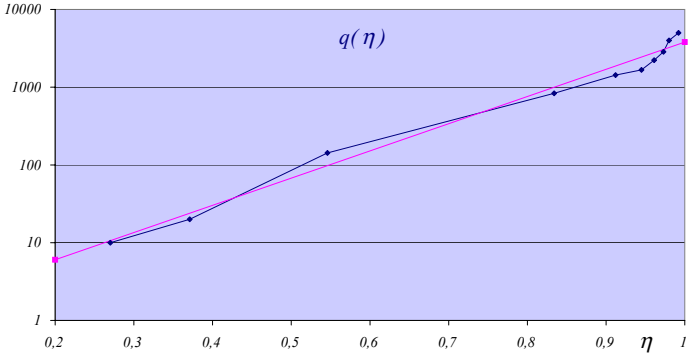


Figure 7: Plot of $q(\eta)$ with a logarithmic regression line corresponding to $10^{3.5\eta+0.08}$.

The choice of the scale factor q is important as it allows to adjust the uniformity, with which the parameter λ navigates from $\mathbf{c}_{simp}(0)$ to $\mathbf{c}_{simp}(1)$. Figure 6 shows the curves $\mathbf{c}_{simp}(0), \mathbf{c}_{simp}(0.1), \ldots, \mathbf{c}_{simp}(1)$ for two different choices of q. In a) a bad value has been chosen such that most λ-curves are close to $\mathbf{c}_{simp}(1)$, whereas in b) the users intuition is matched better with a near uniform distribution of the different λ-curves.

As the used optimization functional is independent of scale, also the factor q should be a constant in terms of scale. The optimal q depends primarily on the relation between the lengths of the reference curves $\mathbf{c}_{simp}(0)$ and $\mathbf{c}_{simp}(1)$, which are used to define the fraction

$$\eta = \frac{L(\mathbf{c}_{simp}(0))}{L(\mathbf{c}_{simp}(1))} \in [0, 1]. \tag{11}$$

η even differs for the different curve segments as can be seen in Figure 6 b). In the more bellied segments, the λ-curves are more attracted by the shortest curve, whereas in the shorter

segments the λ-curves are pulled towards the low-curvature curve. This has to be balanced with a different choice of q for each segment. The dependence of q on η was examined by adjusting q for different ηs by hand. Figure 7 plots the result on a logarithmic scale. The regression line corresponds to the relation

$$q(\eta) = 10^{3.5\eta+0.08}. \tag{12}$$

Both the user specified parameter λ as well as the scale $q(\eta)$ can vary over the curve. η is different for each curve segment S_i between two interpolation constraints $\mathbf{p}_i, \mathbf{p}_{i+1}$. A different η_i is therefore defined for each segment in correspondence with 11. Furthermore, is a more flexible curve design possible if the user can also specify different values λ_i for each segment.

Finally, have the values η_i and λ_i to be interpolated along the curve such that continuous function $\eta(s)$ and $\lambda(s)$ are produced. This is necessary to avoid discontinuities in the functional $F_{simp}(\mathbf{c}, \lambda(s))$, which would destroy the continuity of the minimizing curve. One way to achieve higher continuity is the use of spline basis functions defined over the arc length parameter s. As this would destroy the locality of the user defined λ_i the following local interpolation scheme was used.

A C^k-continuous connection between two functions $f(x \in [0,1])$ and $g(x \in [1,2])$ with $f(1) = g(1)$ at $x = 1$ can most easily be achieved by enforcing the first k derivatives to be zero at $x = 1$, i.e. $f'(1) = 0 = g'(1), \ldots, f^{(k)}(1) = 0 = g^{(k)}(1)$. If $f(x)$ is used for interpolation between λ_0 at $x = 0$ and λ_1 at $x = 1$, the following ansatz can be made:

$$f(x) = \lambda_0(1 - \varphi(x)) + \lambda_1\varphi(x),$$

with the interpolation function $\varphi(x \in [0,1])$, that has to fulfill the following constraints

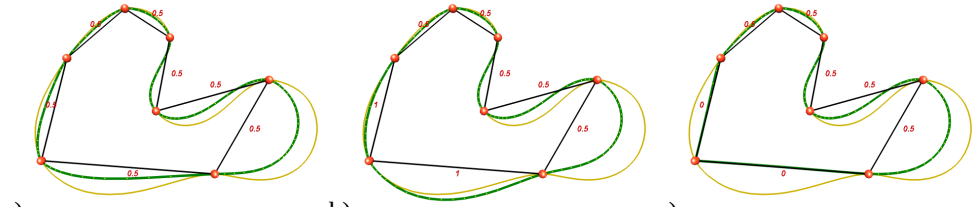
- $\varphi \in [0,1]$, $\varphi(0) = 0$ and $\varphi(1) = 1$
- $\varphi^{(k)}(0) = 0 = \varphi^{(k)}(1)$
- $\varphi(1 - x) = 1 - \varphi(x)$.

The first constraint ensures interpolation of λ_0 and λ_1. The second guarantees a C^k-connection to the previous and next function and the third assures symmetry.

From the first two constraints one can compute for any continuity order k a unique polygonal function $\varphi_k(x)$ of order $2k+1$, which also fulfils the symmetry constraint. The first five of these functions are

$$\begin{aligned}
\varphi_0(x) &= x \\
\varphi_1(x) &= -2x^3 + 3x^2 \\
\varphi_2(x) &= 6x^5 - 15x^4 + 10x^3 \\
\varphi_3(x) &= -20x^7 + 70x^6 - 84x^5 + 35x^4 \\
\varphi_4(x) &= 70x^9 - 315x^8 + 540x^7 - 420x^6 + 126x^5.
\end{aligned}$$

The function φ_4 was used to interpolate λ_i and η_i along the curve. Given the values λ_i and η_i at the edge centers of the polygon $\mathbf{p}_i$, the values at each $\mathbf{p}_i$ were defined to be the averaged values $\bar{\lambda}_i = (\lambda_{i-1} + \lambda_i)/2$ and $\bar{\eta}_i = (\eta_{i-1} + \eta_i)/2$. Each segment S_i was then split in the middle into two parts and in the first half λ was interpolated from $\bar{\lambda}_i$ to λ_i and in the second part from λ_i to $\bar{\lambda}_{i+1}$. The resulting C^4 functions $\lambda(s)$ and $\eta(s)$ where finally plugged into the functional 10 with $q(s) = q(\eta(s))$.



a) b) c)

Figure 8: Curve design with local changes of the parameters λ_i. The adjusted parameters are written close to the edge centers of the polygon $\mathbf{p}_i$. a) all λ_i are the same, b) two λ_i set to one, dragging the curve to $\mathbf{c}_{simp}(1)$, c) the same two λ_i set to zero, dragging the curve to $\mathbf{c}_{simp}(0)$.

Figure 8 gives an example for the design of a curve with different λ_is adjusted. During the curve design the reference curves $\mathbf{c}_{simp}(0)$ (black) and $\mathbf{c}_{simp}(1)$ (yellow) were drawn to guide the user.

3.3 Arbitrarily Smooth Curves

As a second application of the general optimization functional the creation of highly continuous interpolating curves was examined. Figure 9 shows a study of interpolating curves, were the squared k-th derivative of the curvature was minimized for $k = 1, 2, 3, 4$. The implicit side conditions of the Euler-Langrange equation up to the $(k-1)$-th derivative imply that the shown curves are C^2, C^3, C^4 and C^5 continuous.

One of the future goals is to design an interpolating C^∞-curve, probably with all $\alpha_k = 2$ and an appropriate choice for the β_k. For this an analysis similar to the previous section has to be performed in order to find λ-parameters for each β_k. With these λ_ks one can define a C^∞-curve from a sequence $\lambda_0, \lambda_1, ...$

4 Conclusion and Future Work

In this work an optimization algorithm was implemented that allows to compute interpolating curves that minimize a very general functional based on the curve length, curvature and curvature derivatives. A new formula for the computation of the discrete curvature was introduced, that provides a useful gradient for optimization also at arbitrarily sharp corners.

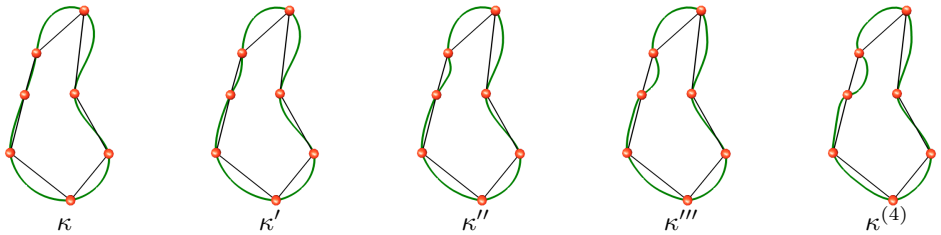


Figure 9: Study of minimizing higher and higher derivatives of the curvature.

The second part of the paper studies the support for additional constraints on the normal of the curve. A functional with one intuitive parameter λ was proposed that allows to trade off between short and low-curvature curves. Finally, the use of a functional with all curvature derivatives was suggested to define a C^∞ interpolating curve.

There are a lot of avenues for future work. The first direction would be a generalization to three dimensions in order to allow camera path planning. Further constraints defined by obstacles could be easily incorporated as long as an initial intersection free path is provided. The second direction of future work would be the generalization to surfaces. There is a lot of ongoing research on how to compute good curvature approximations on discrete surfaces [15, 5, 8]. Surfaces, of which curvature and area can be traded off could provide an important reference to analyze new approaches. Finally, there are a lot of applications in medical imaging and computer vision, where optimal curves and surfaces are placed in a potential field derived from a 2d or 3d image in order to solve the segmentation problem in a robust way.

References

[1] J. H. Ahlberg, E. N. Nielson, and J. L. Walsh. *The Theory of Splines and Their Applications*. Academic Press, New York, 1967.

[2] A. Alon and S. Bergmann. Generic smooth connection functions: a new analytic approach to hermite interpolation. *Journal of Physics A: Mathematical and General*, 35:3877–3898, 2002.

[3] R. Bartels, J. Beatty, and B. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, 1987.

[4] G. Birkhoff and C. R. De Boor. Piecewise polynomial interpolation and approximation. *Approximation of Functions*, pages 164–190, 1965.

[5] F. Cazals and M. Pouget. Estimating differential quantities using polynomial fitting of osculating jets. In *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 177–187. Eurographics Association, 2003.

[6] G. E. Forsythe E. H. Lee. Variational study of nonlinear spline curves. *SIAM Rev.*, 15(i):120–133, 1973.

[7] J. Edwards. Exact equations of the nonlinear spline. *ACM Transactions on Mathematical Software*, 18:174–192, 1992.

[8] J. Goldfeather and V. Interrante. A novel cubic-order algorithm for approximating principal direction vectors. *ACM Trans. Graph.*, 23(1):45–63, 2004.

[9] M. Golomb and J. W. Jerome. Equilibria of the curvature functional and manifolds of nonlinear interpolating spline curves. *SIAM J. Math. Anal.*, 13:421–458, 1982.

[10] B. K. P. Horn. The curve of least energy. *ACM Transactions on Mathematical Software*, 9(4), December 1983.

[11] F. M. Larkin. An interpolation procedure based on fitting elasticas to given data points. Technical Report COS Note No. 5/66, Theory Division, Culham Lab , Abingdon, U K 1966, 1966.

[12] M. Mächler. Very smooth nonparametric curve estimation by penalizing change of curvature. Technical Report 71, ETH Zürich, May 1993.

[13] M. A. Malcolm. On the computation of nonlinear spline functions. *SIAM Journal of Numerical Analysis*, 14(2):254–282, 1977.

[14] E. Mehlum. Curve and surface fitting based on variational criteriae for smoothness. Technical report, Central Institute for Industrial Research, Oslo, 1969.

[15] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In H.-C. Hege and K. Polthier, editors, *Visualization and Mathematics III*, pages 35–58. Springer, 2003.

[16] H. P. Moreton and C. H. Sequin. Functional optimization for fair surface design. In *Proceedings of ACM Siggraph*, pages 167–176, 1992.

[17] K.-D. Reinsch. Numerische berechnung von bmgelinien in der ebene. Technical Report Rep. M8108, Tech. Univ. of Munich, 1981.

[18] R. Schneider and L. Kobbelt. Discrete fairing of curves and surfaces based on linear curvature distribution. In *Proceedings of Curve and Surface Design*, pages 371–380, 1999.

[19] I. J. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions. *Quart. Appl. Math.*, 4:45–99;112–141, 1946.

[20] D. G. Schweikert. An interpolation curve using a spline in tension. *Journal of Mathematics and Physics*, 45:312–317, 1966.

[21] I. Sokolnikoff. *Mathematical Theory of Elasticity*. Krieger Publishing Company, 1983.